PATENT ABSTRACTS OF JAPAN

(11)Publication number : **2000-215080**

(43)Date of publication of application : **04.08.2000**

(51)Int.Cl. G06F 11/28
G06F 15/00
G06F 15/16
G06F 15/177

(21)Application number : **11-013490**

(22)Date of filing : **21.01.1999**

(71)Applicant : **NTT COMMUNICATIONWARE CORP**

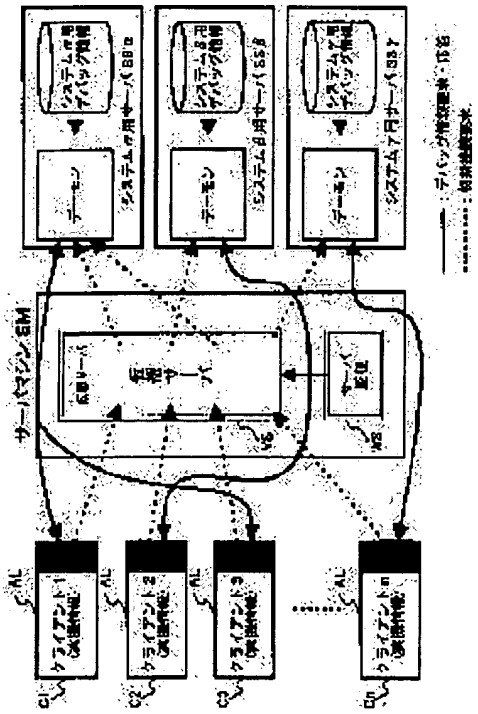
(72)Inventor : **HIRANO MITSUNORI**
YAMADA YOICHI
KAMIHITO MASAAKI
TATE TAKESHI

(54) TERMINAL DEVICE, REPEATING DEVICE, SERVER AND DEBUGGING INFORMATION ACQUISITION SYSTEM, DEBUG INFORMATION ACQUIRING METHOD, AND RECORDING MEDIUM

(57)Abstract:

PROBLEM TO BE SOLVED: To obtain a terminal device, a repeating device, server and debug information acquisition system, debug information acquiring method, and recording medium which can obtain desired symbol information remotely without paying attention to differences of a target debug machine.

SOLUTION: When a client C specifies a desired system, the specified system is connected to a system-adaptive server SS accessible to the debug information of the desired system through a virtual server VS and at a symbol conversion request for a program to be debugged, symbol information is sent from the system-adaptive server SS to the client C.



LEGAL STATUS

[Date of request for examination]

[Date of sending the examiner's decision of rejection]

[Kind of final disposal of application other than the examiner's decision of rejection or application converted registration]

[Date of final disposal for application]

[Patent number]

[Date of registration]

[Number of appeal against examiner's decision of rejection]

[Date of requesting appeal against examiner's decision of rejection]

[Date of extinction of right]

(19)日本国特許庁（ＪＰ）　(12) **公開特許公報（Ａ）**　(11)特許出願公開番号
特開2000－215080
（P2000－215080A）
(43)公開日　平成12年８月４日(2000.8.4)

(51)Int.Cl.⁷		識別記号	FI			テーマコード（参考）
Ｇ０６Ｆ	11/28		Ｇ０６Ｆ	11/28	Ｌ	５Ｂ０４２
	15/00	３２０		15/00	３２０Ｊ	５Ｂ０４５
	15/16	６２０		15/16	６２０Ｂ	５Ｂ０８５
	15/177	６７８		15/177	６７８Ｈ	

審査請求　未請求　請求項の数８　ＯＬ　（全 17 頁）

(21)出願番号　特願平11－13490

(22)出願日　平成11年１月21日(1999.1.21)

(71)出願人　397065480
エヌ・ティ・ティ・コミュニケーションウェア株式会社
東京都港区港南一丁目９番１号
(72)発明者　平野　光徳
東京都港区港南一丁目９番１号　エヌ・ティ・ティ・コミュニケーションウェア株式会社内
(74)代理人　100098084
弁理士　川▲崎▼　研二

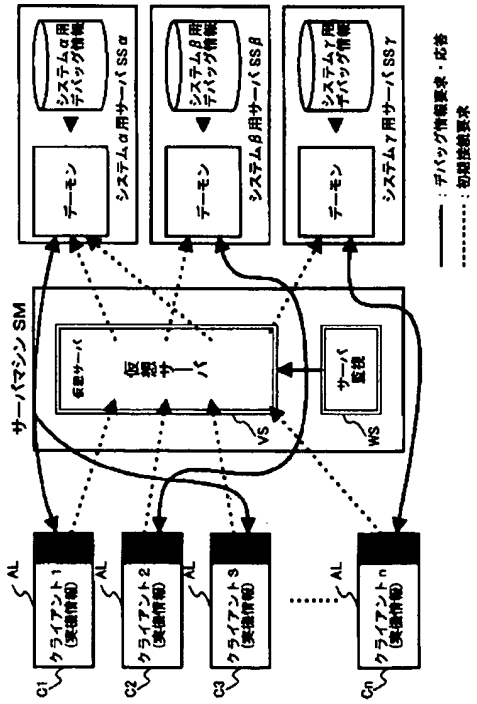
最終頁に続く

(54)【発明の名称】　端末装置、中継装置、サーバならびにデバッグ情報取得システム、デバッグ情報取得方法および記録媒体

(57)【要約】

【課題】　ターゲットとなるデバッグマシンの差異を意識することなく、遠隔で所望のシンボル情報を取得することができる、端末装置、中継装置、サーバならびにデバッグ情報取得システム、デバッグ情報取得方法および記録媒体を提供する。

【解決手段】　クライアントＣから所望のシステムを指定すると、仮想サーバＶＳを介して所望のシステムのデバッグ情報にアクセス可能なシステム対応サーバＳＳと接続され、デバッグ対象プログラムに対するシンボル変換要求に応答して、システム対応サーバＳＳからクライアントＣに向けてシンボル情報が送信される。



【特許請求の範囲】
【請求項１】　異なる複数のシステムのデバッグ情報の内、１または２以上のシステムのデバッグ情報に各々がアクセス可能な複数のサーバと中継装置を介して接続され、前記デバッグ情報の中から所望のシステムのデバッグ情報を取得してプログラムのデバッグが行われる端末装置であって、
回線の接続を要求する旨の信号を前記中継装置に送信する手段と、
この回線接続要求信号に応答して前記中継装置から送信された回線の接続を承認する旨の信号を受信した後、前記複数のシステムの中から所望のシステムを指定して前記中継装置に通知する手段と、
この指定されたシステムのデバッグ情報にアクセス可能なサーバと前記中継装置を介して接続された後、当該サーバに向けてデバッグ対象のプログラムに対するシンボル変換要求を発行する手段と、
このシンボル変換要求に応答して該サーバから送信されたデバッグ情報を受信する手段と、
を備えたことを特徴とする端末装置。
【請求項２】　端末装置と、異なる複数のシステムのデバッグ情報の内、１または２以上のシステムのデバッグ情報に各々がアクセス可能な複数のサーバとの間に介挿され、前記端末装置と前記サーバとを接続する中継装置であって、
前記端末装置からの回線接続要求に応答して回線接続の可否を前記端末装置に通知する手段と、
前記端末装置から指定されたシステムに関する通知を受け取る手段と、
この指定されたシステムのデバッグ情報にアクセス可能なサーバを前記複数のサーバの中から選出して当該サーバへの接続の可否を判定するとともに、該サーバへの接続が可能であると判定した場合に該サーバと前記端末装置とを接続する手段と、
を備えたことを特徴とする中継装置。
【請求項３】　中継装置を介して端末装置と接続され、１または２以上のシステムのデバッグ情報であってシンボル情報を示唆する多数の構成要素からなるデバッグ情報にアクセス可能なサーバであって、
前記端末装置から発行されたデバッグ対象のプログラムに対するシンボル変換の要求に応答して、前記デバッグ情報の中から前記シンボル変換を行うために必要な構成要素を抽出する抽出手段と、
この抽出された構成要素を用いてシンボル情報を作成し、前記端末装置に向けて送信する送信手段と、
を備えたことを特徴とするサーバ。
【請求項４】　請求項３に記載のサーバにおいて、
前記端末装置から発行されたシンボル変換要求を所定のインタフェースを介して受け付ける手段を備え、
前記送信手段は、この所定のインタフェースを介して、作成されたシンボル情報を前記端末装置に向けて送信することを特徴とするサーバ。
【請求項５】　請求項３または４に記載のサーバにおいて、
前記端末装置からのシンボル変換要求に応じたシンボル情報を作成することができない場合には、所定の既定値が格納されたデータを前記端末装置に向けて送信することを特徴とするサーバ。
【請求項６】　請求項１に記載の端末装置と、請求項２に記載の中継装置と、請求項３～５のいずれかに記載のサーバとを備えたことを特徴とするデバッグ情報取得システム。
【請求項７】　異なる複数のシステムのデバッグ情報の内、１または２以上のシステムのデバッグ情報に各々がアクセス可能な複数のサーバのいずれかと端末装置とが中継装置を介して接続され、前記デバッグ情報の中から所望のシステムのデバッグ情報を前記端末装置側で取得するデバッグ情報取得方法であって、
回線の接続を要求する旨の信号を前記端末装置から前記中継装置に送信する過程と、
この回線接続要求信号に応答して前記中継装置から送信された回線の接続を承認する旨の信号を受信した前記端末装置が、前記複数のシステムの中から所望のシステムを指定して前記中継装置に通知する過程と、
この指定されたシステムのデバッグ情報にアクセス可能なサーバと前記中継装置を介して前記端末装置が接続された後、該端末装置から当該サーバに向けてデバッグ対象のプログラムに対するシンボル変換要求を発行する過程と、
このシンボル変換要求に応答して該サーバから送信されたデバッグ情報を前記端末装置側で受信する過程と、
を備えたことを特徴とするデバッグ情報取得方法。
【請求項８】　請求項７に記載の方法を実行するプログラムを記録したことを特徴とする記録媒体。
【発明の詳細な説明】
【０００１】
【発明の属する技術分野】この発明は、シンボリック情報を取得してデバッグ作業を行う際に用いて好適な端末装置、中継装置、サーバならびにデバッグ情報取得システム、デバッグ情報取得方法および記録媒体に関する。
【０００２】
【従来の技術】一般に、開発したソースプログラムの検証に際しては、該ソースプログラムをコンパイラによってコンパイルしてオブジェクトコードに変換し、これをデバッガに実行させることにより設計通りに動作するかどうかを確認する方法が採られる。このデバッガには、ＣやＣ＋＋などの高級言語で書かれたプログラムをソースプログラムのレベルでデバッグできるシンボリックデバッガがある。このシンボリックデバッガでは、デバッグ対象のプログラムに基づき上記高級言語のコンパイラ

が生成・出力する情報（シンボル情報テーブル）やスタックフレーム等が利用され、ユーザがメモリ内容の参照と変更、プログラム動作のトレース、ブレークポイントの設定、変数の値設定、変数の値表示などを高水準言語のレベルで行い、デバッグ対象のプログラムの動作状況を随時確認することができる。

【０００３】特に電話交換機の制御プログラムのような多岐に亘るプロセッサの下で稼働する大規模なソフトウェアの開発においては、このようなシンボリックデバッグはプロセッサ毎に異なるデバッグ情報を取得するために、複数のリモートサーバと複数の端末との間で行われる。図１６は、このようなリモートサーバＲＳ１，ＲＳ２，ＲＳ３とユーザ端末Ｔとからなるデバッグシステムの構成を模式的に示したものである。この場合、各サーバＲＳ１～ＲＳ３はそれぞれ、システムα用のデバッグ情報、システムβ用のデバッグ情報、システムγ用のデバッグ情報を保有し、各端末を操作するユーザが所望のシステムのシンボル情報を取得してデバッグ作業が行われる。

【０００４】

【発明が解決しようとする課題】しかし、このような従来のデバッグシステムでは、ターゲットとするシステムのデバッグマシンごと（図１６に示した例ではシステムα，β，γのデバッグマシンごと）に、デバッグ情報およびそのデバッグ情報を利用するプログラムの両者を揃えることが必要であった。つまり、従来のデバッグ方式では、システムα用のデバッガ、システムβ用のデバッガ、・・・の如く個別にデバッガを作成する必要があった。従って、ターゲットマシンの種類が増えると、そのターゲットマシンにおけるデバッグ情報およびそのデバッグ情報を利用するためのプログラムの双方を備えることが必要であった。

【０００５】また、実際のデバッグ作業の際には、所望のデバッグ情報を保有するマシンにリモートでログインしなければならなかった。従って、ターゲットとなるマシンのアドレス（例えばＩＰアドレス）やホスト名、デバッグ情報ファイルやデバッグ情報を利用するプログラムの存在場所（ディスク内のパス名やファイル名）を予めユーザが認識している必要があった。そして、別のターゲットマシンに対してデバッグ作業を試みるには、ユーザが変更するマシンを意識しなければならないといった状況があった。さらに、特定のマシンに対するデバッグ作業に複数のユーザが集中すると、ホストマシンの負荷が増大し、デバッグ効率が悪くなるといった点も指摘されていた。

【０００６】そこで、この発明は、このような従来の課題に着目してなされたもので、ターゲットとなるデバッグマシンの差異を意識することなく、遠隔端末から所望のシンボリック情報を取得してデバッグ作業を行うことができる端末装置、中継装置、サーバ、ならびにデバッグ情報取得システム、デバッグ情報取得方法および記録媒体を提供することを目的とするものである。

【０００７】

【課題を解決するための手段】上記目的を達成するために、第１発明の端末装置は、異なる複数のシステムのデバッグ情報の内、１または２以上のシステムのデバッグ情報に各々がアクセス可能な複数のサーバと中継装置を介して接続され、前記デバッグ情報の中から所望のシステムのデバッグ情報を取得してプログラムのデバッグが行われる端末装置であって、回線の接続を要求する旨の信号を前記中継装置に送信する手段と、この回線接続要求信号に応答して前記中継装置から送信された回線の接続を承認する旨の信号を受信した後、前記複数のシステムの中から所望のシステムを指定して前記中継装置に通知する手段と、この指定されたシステムのデバッグ情報にアクセス可能なサーバと前記中継装置を介して接続された後、当該サーバに向けてデバッグ対象のプログラムに対するシンボル変換要求を発行する手段と、このシンボル変換要求に応答して該サーバから送信されたデバッグ情報を受信する手段とを備えたことを特徴とするものである。

【０００８】第２発明の中継装置は、端末装置と、異なる複数のシステムのデバッグ情報の内、１または２以上のシステムのデバッグ情報に各々がアクセス可能な複数のサーバとの間に介挿され、前記端末装置と前記サーバとを接続する中継装置であって、前記端末装置からの回線接続要求に応答して回線接続の可否を前記端末装置に通知する手段と、前記端末装置から指定されたシステムに関する通知を受け取る手段と、この指定されたシステムのデバッグ情報にアクセス可能なサーバを前記複数のサーバの中から選出して当該サーバへの接続の可否を判定するとともに、該サーバへの接続が可能であると判定した場合に該サーバと前記端末装置とを接続する手段とを備えたことを特徴とするものである。

【０００９】第３発明のサーバは、中継装置を介して端末装置と接続され、１または２以上のシステムのデバッグ情報であってシンボル情報を示唆する多数の構成要素からなるデバッグ情報にアクセス可能なサーバであって、前記端末装置から発行されたデバッグ対象のプログラムに対するシンボル変換の要求に応答して、前記デバッグ情報の中から前記シンボル変換を行うために必要な構成要素を抽出する抽出手段と、この抽出された構成要素を用いてシンボル情報を作成し、前記端末装置に向けて送信する送信手段とを備えたことを特徴とするものである。

【００１０】第４発明のサーバは、上記第３発明のサーバにおいて、前記端末装置から発行されたシンボル変換要求を所定のインタフェースを介して受け付ける手段を備え、前記送信手段は、この所定のインタフェースを介して、作成されたシンボル情報を前記端末装置に向けて

送信することを特徴とするものである。

【００１１】第５発明のサーバは、上記第３または第４発明のサーバにおいて、前記端末装置からのシンボル変換要求に応じたシンボル情報を作成することができない場合には、所定の既定値が格納されたデータを前記端末装置に向けて送信することを特徴とするものである。

【００１２】第６発明のデバッグ情報取得システムは、上記第１発明の端末装置と、上記第２発明の中継装置と、上記第３～第５発明のいずれかのサーバとを備えたことを特徴とするものである。

【００１３】第７発明のデバッグ情報取得方法は、異なる複数のシステムのデバッグ情報の内、１または２以上のシステムのデバッグ情報に各々がアクセス可能な複数のサーバのいずれかと端末装置とが中継装置を介して接続され、前記デバッグ情報の中から所望のシステムのデバッグ情報を前記端末装置側で取得するデバッグ情報取得方法であって、回線の接続を要求する旨の信号を前記端末装置から前記中継装置に送信する過程と、この回線接続要求信号に応答して前記中継装置から送信された回線の接続を承認する旨の信号を受信した前記端末装置が、前記複数のシステムの中から所望のシステムを指定して前記中継装置に通知する過程と、この指定されたシステムのデバッグ情報にアクセス可能なサーバと前記中継装置を介して前記端末装置が接続された後、該端末装置から当該サーバに向けてデバッグ対象のプログラムに対するシンボル変換要求を発行する過程と、このシンボル変換要求に応答して該サーバから送信されたデバッグ情報を前記端末装置側で受信する過程とを備えたことを特徴とするものである。

【００１４】第８発明の記録媒体は、上記第７発明の方法を実行するプログラムを記録したことを特徴とするものである。なお、このプログラムをＣＤ－ＲＯＭ、テープメディア、ＤＶＤ－ＲＡＭ等の記録媒体に格納して頒布する（ネットワーク経由でこのプログラムを配信したり、ダウンロードする行為も含む）ことによっても本発明を実施することができる。

【００１５】

【発明の実施の形態】この発明の好ましい実施の形態について、以下、添付図面を参照しつつ詳細に説明する。

【００１６】１．実施形態の構成

１．１．基本構成

図１は、本実施形態にかかるデバッグシステムの概要を示したものであり、図２は、このデバッグシステムのより具体的な構成について示したものである。このデバッグシステムは、デバッグ用端末であるｎ台のクライアントＣ１～Ｃｎ（任意のクライアントを示す場合には「クライアントＣ」と表記するものとする）と、仮想サーバＶＳと監視用サーバＷＳとからなるサーバマシンＳＭと、それぞれ異なるＣＰＵ（α，β，γ，・・・・）固有のデバッグ情報を保有するシステム対応サーバＳＳα，ＳＳβ，ＳＳγ，・・・・（これらのシステム対応サーバの内、任意のシステム対応サーバを示す場合には「システム対応サーバＳＳ」と表記するものとする）とで構成され、これらの装置は伝送路となるイーサネットＥＮＥＴによって相互に接続されている。

【００１７】１．２．クライアントおよびサーバマシンの構成

クライアントＣは、ユーザがデバッグ作業を行うための端末装置である。このクライアントＣからシステム対応サーバＳＳの記憶装置内に記憶されているソースプログラムを呼び出してデバッグ指令を与え、シンボル変換されたデバッグ情報（シンボリック情報）をシステム対応サーバＳＳから取得する。そして、このデバッグ情報を参照してソースプログラムの検証が行われる。

【００１８】この際、クライアントＣ側から所望のシステムのデバッグ情報を取得したい旨の信号が送出されると、仮想サーバＶＳを介してこの所望のシステムのデバッグ情報にアクセス可能なシステム対応サーバＳＳと自動的に接続される。そのために、クライアントＣは、仮想サーバＶＳおよびシステム対応サーバＳＳへのアクセスインタフェースとしての機能を果たすライブラリＡＬを保有している。そして、このシステム対応サーバＳＳとクライアントＣとの間でデバッグ情報の要求およびその要求に対する応答がなされ、デバッグ情報が供給される仕組みになっている。そして、一連のデバッグ情報が供給されると、クライアントＣはシステム対応サーバＳＳに対して回線切断要求信号を発行する。一方、システム対応サーバＳＳは、クライアント側から要求された作業内容を要約したログを仮想サーバＶＳへ格納すべく、仮想サーバＶＳにログ格納を依頼する信号を送出する。

【００１９】仮想サーバＶＳは、クライアントＣからの接続要求を受け付け、システム対応サーバＳＳのデーモンプロセス（例えば、ＵＮＩＸ上のｉｎｅｔｄプロセス）に働きかけてクライアントＣとシステム対応サーバＳＳとを接続する接続用プロセスを生成し、各システム対応サーバとの通信を行う。そして、クライアントＣとシステム対応サーバＳＳとが接続され、デバッグ情報がクライアントＣに供給された後、クライアントＣからの回線切断要求を受け付け、その旨のメッセージ信号をシステム対応サーバＳＳに送信し、回線を切断する。このように、仮想サーバＶＳは中継装置としての機能を有する。

【００２０】監視サーバＷＳは、各システム対応サーバＳＳのプロセスを監視し、仮想サーバＶＳに監視結果を随時通知する。これを受けて仮想サーバＶＳは、システムの負荷分散が図られるようにシステム対応サーバＳＳの振り分けを行ってクライアントＣとシステム対応サーバＳＳとの接続を行う。また、監視サーバＷＳは、仮想サーバＶＳのプロセスについても監視を行っており、仮想サーバＶＳのプロセスに異常が発見された場合、プロ

セスを再起動し、仮想サーバＶＳに接続中の全接続プロセスを再構築するようになっている。

【００２１】１．３．システム対応サーバの構成

システム対応サーバＳＳとして、図１に示したように、システムα用デバッグ情報にアクセス可能なシステムα用サーバＳＳα、システムβ用デバッグ情報にアクセス可能なシステムβ用サーバＳＳβといった具合に１種類のデバッグ情報にのみアクセス可能なタイプがある。また、図２に示したように、システムγ用デバッグ情報およびシステムθ用デバッグ情報のいずれかにアクセス可能なシステムγθ用サーバＳＳγθといった具合に複数種類のデバッグ情報のうちの１つにアクセス可能なタイプもある。そして、これらのタイプのサーバが適宜分散して配置されている。さらに、ネットワークを介して外部からの接続を可能とすべく、デーモンプロセスが起動されている。

【００２２】このシステム対応サーバＳＳは、膨大な交換機用のプログラムを保有している。つまり、ソースプログラム、オブジェクトモジュール、実行モジュールを含めた膨大な数のファイルを記憶装置内に記憶している（これらのファイルの詳細については後述する）。そして、クライアントＣからのシンボル変換要求信号を受信してシンボル変換処理を行い、その結果、シンボル情報をクライアントＣに送信する。

【００２３】また、システム対応サーバＳＳは、クライアントＣからのデバッグ情報取得要求を受けて、汎用フォーマットであるＥＬＦ(Executable and Linkable Format)／ＤＷＡＲＦ(Debug With Arbitrary Record Format)に対応した構成要素へのアクセスがなされたものとみなし、内部ではＥＬＦ／ＤＷＡＲＦ以外のファイルフォーマットに対応した情報要素へアクセスし、シンボル情報を抽出した上でＥＬＦ／ＤＷＡＲＦ形式に変換してクライアントＣに結果を返す。

【００２４】１．３．１．システム対応サーバで取り扱われるファイルフォーマットとデバッグフォーマット

システム対応サーバＳＳは、既に説明したようにソースファイルや実行形式ファイルを含む交換機用のプログラムに関連する一連の膨大なファイルを保有し、ユーザが指定したプログラムに対するデバッグ結果としてのデバッグ情報をクライアント側に供給する。そこで、このファイルやデバッグ情報のフォーマットについて説明する。

【００２５】ファイルフォーマットは、ＯＳがプログラムを実行するために規定された実行形式ファイルの構成を定めたものである。また、デバッグフォーマットは、コンパイラがソースファイルと実行形式ファイルの要素間（ソースの定義と割り付けアドレス等）の対応を生成した情報一式の構成を定めたものであり、主としてデバッガがソースレベルのデバッグを行う際に使用される。ファイルフォーマットとデバッグフォーマットとは、プログラムの構成において対で使用されるものである。

【００２６】図８は、このファイルフォーマットとデバッグフォーマットとの関係について示している。同図（ａ）に示したように、ソースファイル１をコンパイラ２によってコンパイルすることにより、ファイルフォーマットに準拠した実行形式ファイル３が生成される。一方、コンパイル時にデバッグ情報を生成するよう指示が与えられると、デバッグフォーマットに準拠したデバッグ情報４が実行形式ファイル３の一部として（これをセッションという）付加される。

【００２７】同図（ｂ）に示したように、ファイルフォーマットとデバッグフォーマットとの対応関係はＣＰＵごとに異なったものとなっている。すなわち、ＣＰＵαを搭載したプラットフォームでは、ファイルフォーマット、デバッグフォーマットともにＳＹＳＲＯＦが用いられる。また、ＣＰＵβを搭載したプラットフォームでは、ファイルフォーマットとしてＥＬＦ、デバッグフォーマットとしてＥＷＳが用いられ、ＣＰＵγを搭載したプラットホームではファイルフォーマットとしてＥＬＦ、デバッグフォーマットとしてＤＷＡＲＦが用いられる。

【００２８】図５は、これら各種フォーマットを利用するためのシステム対応シンボリックサーバＳＳの機能構成について示したものである。このように、システム対応シンボリックサーバＳＳは、クライアントＣからのデバッグ情報取得要求に応じて、ＣＰＵα用ＯＳ個別部Ｅα，ＣＰＵβ用ＯＳ個別部Ｅβ，ＣＰＵγ用ＯＳ個別部Ｅγのいずれかを介して、ファイルフォーマットアクセスモジュールＦＡＭ，デバッグフォーマットアクセスモジュールＤＡＭにアクセスして必要な情報を取得し、クライアント側に供給する。

【００２９】すなわち、クライアントＣ側からＣＰＵα用ＯＳに対するデバッグ情報の取得要求が発行されると、システム対応サーバＳＳはＣＰＵα用ＯＳ個別部Ｅαを動作させて、ファイルフォーマットアクセスモジュールＦＡＭおよびデバッグフォーマットアクセスモジュールＤＡＭとして、ともにＳＹＳＲＯＦ形式のデータにアクセスする。

【００３０】同様に、クライアントＣ側からＣＰＵβ用ＯＳに対するデバッグ情報の取得要求が発行されると、システム対応サーバＳＳはＣＰＵβ用ＯＳ個別部Ｅβを動作させて、ファイルフォーマットアクセスモジュールＦＡＭとして、ＥＬＦ形式のデータに、デバッグフォーマットアクセスモジュールＤＡＭとして、ＥＷＳ形式のデータにアクセスする。

【００３１】さらに、クライアントＣ側からＣＰＵγ用ＯＳに対するデバッグ情報の取得要求が発行されると、システム対応サーバＳＳはＣＰＵγ用ＯＳ個別部Ｅγを動作させて、ファイルフォーマットアクセスモジュールＦＡＭとして、ＥＬＦ形式のデータに、デバッグフォー

マットアクセスモジュールＤＡＭとして、ＤＷＡＲＦ形式のデータにアクセスする。

【００３２】そして、システム対応サーバＳＳは、このようにアクセスしたデータの中からシンボル変換を行うために必要なデータを抽出してシンボル情報を作成し、クライアントＣ側に供給する。

【００３３】２．実施形態の動作

２．１．基本動作シーケンス

図３は、このようなデバッグシステムにおける基本的な動作シーケンスについて示している。まず、クライアントＣのユーザからデバッグ開始指令が与えられると、クライアントＣから仮想サーバＶＳに向けて回線の接続を要求する旨の信号（初期接続要求信号）が送出される（Ｓ１）。仮想サーバＶＳは各システム対応サーバＳＳに接続しているユーザ数やプロセスの状況を判断して、回線の接続を承認する旨の信号をクライアントＣに送出する（Ｓ２）。

【００３４】次に、クライアントＣからデバッグ対象のシステム名（α，β，γ，・・・）が指定される（Ｓ３）。これを受けて、仮想サーバＶＳは指定されたシステムのデバッグ情報を保有するシステム対応サーバＳＳが起動されているか否かの確認作業を行う（Ｓ４）。指定されたシステムのデバッグ情報を保有するサーバＳＳの起動が確認されると（Ｓ５）、その旨のメッセージ信号とともにこのシステム対応サーバＳＳの名称と通信ポートとが指定される（Ｓ６）。

【００３５】クライアントＣからデバッグ対象のプログラムに対するシンボル変換を要求する旨の信号が送出されると（Ｓ７）、これを受けたシステム対応サーバＳＳからシンボル情報がクライアントＣに返送される（Ｓ８）。そして、このステップＳ７とステップＳ８の一連の動作に関するログ情報が仮想サーバＶＳにストックされる（Ｓ９）。ソースプログラムのデバッグ過程において、上記ステップＳ７～Ｓ９の動作が繰り返し行われ、一連のデバッグ情報がクライアントＣに供給される。

【００３６】一通りユーザの所望するデバッグ情報が取得されると、クライアントＣから回線の切断要求が発行される（Ｓ１０）。システム対応サーバＳＳから回線の切断要求が承認されると（Ｓ１１）、一連のデバッグ情報取得シーケンスが終了する。これらの動作シーケンスに対するログ情報は仮想サーバＶＳにストックされる（Ｓ１２）。

【００３７】２．２．ユーザ側の操作を中心とした動作シーケンス

今度は視点を変えて、デバッグ作業時におけるユーザ側の操作を中心とした動作シーケンスについて、図４を用いて説明する。同図は、クライアントＣ側のソースレベルデバッガの操作画面におけるユーザ側のオペレーション、仮想サーバＶＳ、システムα用シンボリック対応サーバＳＳα、ソースレベルデバッガ実行部の取り扱い対象となり、システムαをターゲットとしてコンパイルされたシステムα用ロードモジュールＬＭαに関し、これら相互間のやり取りを示したシーケンス図である。

【００３８】まず、クライアントＣ側からシンボリック対応サーバＳＳαの利用を予約するための信号が送出される（Ｓ２０）。これを受けて、仮想サーバＶＳは新たにシンボリック対応サーバＳＳαの回線接続用のプロセスを起動して回線を接続するよう指示を与える（Ｓ２１）。そして、仮想サーバＶＳは、ネットワークからのステータスを元にシンボリックサーバＳＳαの接続情報を返却して（Ｓ２２）、クライアントＣとシンボリック対応サーバＳＳαとが接続される。このようにして、一旦、ターゲットデバッグマシンであるシンボリック対応サーバＳＳαにクライアントＣが接続されると、以後の動作では仮想サーバＶＳを介さず両者の間で直接信号やデータのやり取りが行われる。

【００３９】次に、クライアントＣからシンボリック対応サーバＳＳα内のディスクに記憶され、デバッグ情報を含む多数のファイル（構成ファイルという）の一覧の取得を要求する信号が送出される（Ｓ２３）。構成ファイルの一覧が取得できたら、クライアントＣのユーザは、この構成ファイルの中から所望のソースファイルをオープンし（Ｓ２４）、ブレークポイントを設定する（Ｓ２５）。

【００４０】そして、このソースファイルの行番号からマイクロプロセッサが実行しうる機械語のアドレスを取得するための行番号アドレス変換指令が、システムα用シンボリックサーバＳＳαに向けて発行される（Ｓ２６）。シンボリックサーバＳＳαは、ロードモジュールＬＭαの中から行番号アドレス変換情報を抽出して、クライアントＣにアドレス情報を返却する（Ｓ２７）。なお、この「行番号アドレス変換」の仕組みについては後述する。

【００４１】続いて、ユーザの設定したブレークポイントに関する情報が、ロードモジュールＬＭαに対して設定され、プログラムの先頭からブレークポイントに至るまで実行するよう指示が与えられる（Ｓ２９）。そして、ブレークポイントまでプログラムが実行されたらデバッガに制御を戻してプログラムの実行を停止し、その旨がクライアントＣ側に通知される（Ｓ３０）。このように、通常にプログラムを実行させ、ブレイクポイントに到達した時点でデバッガに制御を移行させる方法は、ブレイクポイントで指定した番地のマシン語をイリーガルファンクション（ＩＦ）と呼ばれる未定義命令に置き換え、このＩＦを実行することで、ＣＰＵＨａｌｔを発生させてデバッガに制御を移し、ＩＦを元の命令に書き戻して１ステップ実行させ、またＩＦを書き込むという動作によって実現される。

【００４２】さらに、ユーザはデバッガ操作画面上でソースプログラム内の変数を参照し（Ｓ３１）、変数アド

レス変換指令をシステムα用サーバＳＳαに与える（Ｓ３２）。システムα用サーバＳＳαは、ロードモジュールＬＭαから変数アドレス情報を抽出して、ソースプログラム内で参照された変数のアドレスをクライアントＣ側に返却する（Ｓ３３）。

【００４３】次に、クライアントＣ側からこの変数の値を参照するようロードモジュールＬＭαに指令を与える（Ｓ３４）。すると、ロードモジュールＬＭαからこの変数の値が返却される（Ｓ３５）。

【００４４】続いて、クライアントＣは、ソースレベルデバッガの操作画面において、１ステップずつプログラムをトレースして実行させる指令を与える（Ｓ３６）。そして、この指令がロードモジュールＬＭα側に伝送され、１ステップずつプログラムが実行される（Ｓ３７）。

【００４５】その他、種々のデバッグのための操作がデバッガ操作画面からユーザよりなされて（Ｓ３８）、クライアント側の一連のデバッグ作業が終了する（Ｓ３９）。そして、シンボリックサーバＳＳαに対する利用を終了する旨の信号が仮想サーバＶＳに送信され（Ｓ４０）、クライアントＣとの回線を切断するようシンボリックサーバ側に指示が与えられる（Ｓ４１）。最後に仮想サーバＶＳからクライアントＣ側へシンボリックサーバＳＳαとの回線が切断され利用が終了した旨の通知がなされる（Ｓ４２）。

【００４６】２．３．システム対応サーバにおける動作シーケンス

続いて、システム対応サーバＳＳにおける動作シーケンスについて、特にシステムごとに異なるデバッグ情報の構成の差異を隠蔽するための仕組みに焦点をあてて説明する。

【００４７】システム対応サーバＳＳは、各種ファイルフォーマットやデバッグフォーマットの情報要素を、共通フォーマットであるＥＬＦファイルフォーマットとＤＷＡＲＦデバッグフォーマットの情報要素を用いてアクセスし、クライアント側にデバッグ情報を供給する。

【００４８】そこで、まず、このＥＬＦ，ＤＷＡＲＦにおける各情報要素間の依存関係について説明する。図９は、ＥＬＦファイルフォーマットの構成を示している。このＥＬＦファイルフォーマットは、ＥＬＦヘッダＥＬ１と、プログラムヘッダＥＬ２と、セクションＳＥＣ１，セクションＳＥＣ２，・・・，セクションＳＥＣｎ，・・・からなるセクションと、これらのセクションに関する情報を集約したセクションヘッダテーブルＥＬ４とからなる。

【００４９】図１０は、ＥＬＦファイルフォーマットに関し、これら各情報要素の依存関係について示したものである。このファイルフォーマット内には、オブジェクトファイルの種類やプロセッサの種類等を規定したＥＬＦヘッダＥＬ１、実行形式ファイルにおけるセグメントに関する情報等が記録されたプログラムヘッダテーブルＥＬ２、およびセクション名文字列テーブルＥＬ３、セクションヘッダテーブルＥＬ４が格納されている。ＥＬ５～ＥＬ３１は、ＥＬＦファイルフォーマットとして種々の拡張子を有するファイルを示している。そして、図９に示したセクション１～セクションｎ，・・・のそれぞれに対応して、ＥＬ５～ＥＬ３１までの各セクションが格納されている。

【００５０】図１１は、ＤＷＡＲＦファイルフォーマットにおける各情報要素の依存関係について示したものである。このファイルフォーマット内には、.debugセクションＥＬ９にリンクする形でユニット情報ＤＷ１、ソースファイル情報ＤＷ２、・・・・が存在し、以下インライン関数情報ＤＷ３５に至るまで各種情報が細分化されて格納され、シンボリックデバッグ情報が提供される（ＤＷ１～ＤＷ３５はそれぞれＤＷＡＲＦデバッグ情報である）。このように、デバッグ情報は実行形式ファイルの一部として実装されるため、デバッグ情報へアクセスするためには実行形式ファイルの先頭から順次、依存関係を手繰っていくことが必要である。

【００５１】そこで、図４のステップＳ２６に示した「行番号アドレス変換」において、シンボリックサーバＳＳがクライアントＣから要求を受けた後の内部動作を、既に図５において説明した対応するＣＰＵごとに異なる個別部の動作を中心に説明する。

【００５２】２．３．１．行番号アドレス変換の際のシステム対応サーバ側の動作

図６は、上記行番号アドレス変換の詳細を示したフローチャートである。まず、シンボリックサーバＳＳから行番号アドレス変換の指令がＣＰＵγに対応したシンボリックサーバ個別部Ｅγに通知される（Ｓ５０）。この指令を受けたシンボリックサーバ個別部Ｅγは、ＥＬＦファイルフォーマットアクセス部（単にファイルフォーマットアクセス部という）ＦＡにデバッグセクションの取得を要求する（Ｓ５１）。そしてファイルフォーマットアクセス部ＦＡは、ＥＬＦファイル内のＥＬＦヘッダＥＬ１を取得する（Ｓ５２）。

【００５３】次に、ファイルフォーマットアクセス部ＦＡは、セクション名文字列テーブルを抽出する動作を行う（Ｓ５３）。この際、ファイルフォーマットアクセス部ＦＡは、ＥＬＦファイルの各構成要素の中からセクション名文字列テーブルＥＬ３を取得する（Ｓ５４）。さらに、ＥＬＦファイルフォーマットアクセス部ＦＡはデバッグセクション名取得のための要求をＤＷＡＲＦデバッグフォーマットアクセス部（単にデバッグフォーマットアクセス部という）ＤＡに与える（Ｓ５５）。

【００５４】続いて、ファイルフォーマットアクセス部ＦＡは、ＥＬＦファイルの中の.debugセクションを抽出する動作に移行する（Ｓ５６）。この過程でファイルフォーマットアクセス部ＦＡは、実際にセクションヘッダ

テーブルＥＬ１４を取得する（Ｓ５７）。

【００５５】次に、ファイルフォーマットアクセス部ＦＡは、ＥＬＦファイルの中の.debugセクションエントリを抽出する動作に移行する（Ｓ５８）。この過程で、.debugセクションを取得する（Ｓ５９）。

【００５６】そして、シンボリックサーバ個別部Ｅγからデバッグフォーマットアクセス部ＤＡに向けて、ユニット情報を取得するための要求が発行される（Ｓ６０）。これを受けて、デバッグフォーマットアクセス部ＤＡは、ＤＷＡＲＦデバッグ情報内のユニット情報を取得する（Ｓ６１）。そして、デバッグフォーマットアクセス部ＤＡは、このユニット情報内に記録された情報要素を抽出する（Ｓ６２）。

【００５７】続いて、シンボリックサーバ個別部Ｅｒから行番号に対応するアドレスを取得するための要求がデバッグフォーマットアクセス部に向けて発行される（Ｓ６３）。これを受けて、デバッグフォーマットアクセス部ＤＡは、ＤＷＡＲＦデバッグ情報内からソースファイル情報ＤＷ２を取得し、このソースファイル情報ＤＷ２内のデータを参照して行番号に対応したアドレスを取得する（Ｓ６５）。

【００５８】このように、シンボリックサーバは、ユーザからのシンボル変換要求（この例では、「行番号アドレス変換」）を受け付け、この要求を自身が解析可能なファイルフォーマットとデバッグフォーマットのアクセス部ＤＡに向けてＥＬＦ／ＤＷＡＲＦ情報要素と見立てた要求（この例では、「デバッグセクション取得」，「ユニット情報取得」，「行番号対応アドレス取得」）に変換して内部的なシンボル情報の抽出処理を行い、共通フォーマットであるＥＬＦ／ＤＷＡＲＦ形式に変換してクライアントＣ側に供給する。以上のようにして、システム対応シンボリックサーバＳＳはＣＰＵごとに異なるデバッグ情報の差異を隠蔽する機能を有している。

【００５９】２．３．２．未定義情報に対する要求が発行された場合のシステム対応サーバ側の動作

一般に、各ファイルフォーマット／デバッグフォーマットは、それぞれに相関がないため、あるフォーマットに規定されている情報要素が他のフォーマットでは規定されていない場合がある。そこで、これらの齟齬を吸収するために、シンボリックサーバＳＳは各フォーマットアクセス部に規定されていない情報要素へのアクセスが発生した場合に、解析を行うことなく既定値を返却する機構を有する。

【００６０】図７は、ファイルフォーマットおよびデバッグフォーマット内に定義されていない情報に対するアクセスがなされた場合の動作について示している。まず、シンボリックサーバＳＳはユーザ側からの要求に対応する情報がファイルフォーマットおよびデバッグフォーマット内に定義されているか否かの問い合わせを行う（Ｓ７０）。これを受けてシンボリックサーバ個別部Ｅは、ファイルフォーマットアクセス部に問い合わせの処理を行う（Ｓ７１）。そして、ファイルフォーマットアクセス部は、ファイルフォーマットＡ対応の未定義情報要素にアクセスして、未定義情報要素であるか否かの確認を行う（Ｓ７２）。

【００６１】続いて、シンボリックサーバ個別部Ｅは、デバッグフォーマットアクセス部に問い合わせの処理を行う（Ｓ７３）。これを受けて、デバッグフォーマットアクセス部は、デバッグフォーマットＡ対応の未定義情報要素にアクセスして、未定義情報要素であるか否かの確認を行う（Ｓ７４）。このように、ファイルフォーマットおよびデバッグフォーマット内に定義されていない情報に対するアクセスがなされた場合には、所定の既定値をクライアント側に返すことで対応する。

【００６２】２．４．ローカル変数の型名取得の際の動作

それでは、実際に図１２に示したＣ言語によるソースプログラム（ファイル名test.c）のローカル変数ｉの型名取得の際の動作を図１３～図１５を用いて説明する。図１３は、システム対応サーバＳＳにおいて、クライアントＣからのデバッグ情報取得要求を受け付ける際の動作について示したものである。

【００６３】まず、クライアントＣからこのソースプログラムtest.cのmain関数内のローカル変数ｉの型を解決すべき旨の要求が発行される（Ｓ８０）。これを受けてシステム対応サーバＳＳはデバッグ情報アクセス部ＤＡにユニット情報（ソースファイル情報）取得のための要求を発行する（Ｓ８１）。そして、デバッグ情報アクセス部ＤＡからユニット情報が返却される（Ｓ８２）。

【００６４】次に、システム対応サーバＳＳはＰＲＯＣ情報（グローバル関数情報）取得のための要求をデバッグ情報アクセス部ＤＡに向けて発行する（Ｓ８３）。これを受けてＰＲＯＣ情報がデバッグ情報アクセス部ＤＡからシステム対応サーバＳＳに返却される（Ｓ８４）。さらに、変数情報としてのローカル変数情報の取得要求に応答して、ローカル変数情報がデバッグ情報アクセス部ＤＡからシステム対応サーバＳＳに返却される（Ｓ８５，Ｓ８６）。このようにして、変数ｉの型が整数型のｉｎｔである旨の通知がシステム対応サーバＳＳからクライアントＣに向けてなされる（Ｓ８７）。

【００６５】２．４．１．ＤＷＡＲＦにおけるデバッグ情報取得の際の動作

このようなソースプログラムの型解決処理において、デバッガＤＧとデバッグ情報アクセス部ＤＡとの間では図１４に示したような動作シーケンスが実行される。同図では、ユニット情報（ソースファイル）、ＰＲＯＣ情報（グローバル関数情報）、変数情報（ローカル変数情報）のそれぞれについて、デバッグ情報アクセス部ＤＡから返却されるデータの構成をシーケンス図中に明示している。

【００６６】これらＤＷＡＲＦにおけるデバッグ情報（シンボルテーブル）の一要素であるユニット情報、ＰＲＯＣ情報、変数情報のそれぞれの詳細なデータ構成については、図１５（ａ）～（ｃ）に示している。なお、これら情報要素のＤＷＡＲＦデータの体系における位置付けについては、既に説明した図１１に示した（ユニット情報ＤＷ１，グローバル関数情報ＤＷ９，ローカル関数情報ＤＷ１４）通りである。

【００６７】図１５（ａ）に示したように、ユニット情報は、上位バイトから下位バイトに向かって、ユニット名（ｎ＋１）バイト、使用言語４バイト、ユニット開始アドレス４バイト、ユニット終了アドレス４バイト、行番号情報オフセット４バイト、コンパイルディレクトリ（ｎ＋１）バイト、作者名（ｎ＋１）バイトの順で配置された合計３ｎ＋１９バイトのデータ列ＤＬ１によって構成される。同様にＰＯＲＣ情報、および変数情報の場合もそれぞれ図１５（ｂ），（ｃ）の表に示した属性要素の順で配置されたデータ列ＤＬ２，ＤＬ３によって構成されている。

【００６８】図１４に示したユニット情報取得要求（Ｓ８１）とユニット情報返却（Ｓ８２）における処理では、デバッグ情報アクセス部ＤＡからデータ列ＤＬ１からなるユニット情報がデバッガＤＧに返却される。従って、デバッガＤＧ側ではデータ列ＤＬ１の上位（ｎ＋１）バイトに記録された内容を読み取り、test.cというソースファイル名を取得する。

【００６９】次に、ＰＲＯＣ情報取得要求（Ｓ８３）とＰＲＯＣ情報返却（Ｓ８４）における処理では、デバッグ情報アクセス部ＤＡからデータ列ＤＬ２からなるユニット情報がデバッガＤＧに返却される。従って、デバッガＤＧ側ではデータ列ＤＬ２の上位（ｎ＋１）バイトに記録された内容を読み取り、mainというグローバル関数名を取得する。

【００７０】さらに変数情報取得要求（Ｓ８５）と変数情報返却（Ｓ８６）における処理では、デバッグ情報アクセス部ＤＡからデータ列ＤＬ３からなるユニット情報がデバッガＤＧに返却される。従って、デバッガＤＧ側ではデータ列ＤＬ３の上位（ｎ＋１）バイトに記録された内容を読み取り、intというローカル変数の型名を取得し、「ローカル変数ｉの型がｉｎｔである」というシンボリック情報を取得することができる。

【００７１】３．実施形態の効果

（１）リモートログインせずにローカル端末でデバッグ作業をすることができ、仮想サーバがターゲットシステム専用サーバに振り替えるところから、マシンの負荷分散を図ることができる。そして、この仮想サーバの振り替え機能により、クライアントは多くのサーバを意識することなく、仮想サーバだけを認識するだけでシンボリック情報を取得してデバッグ作業を行うことができる。

（２）クライアント側のデバッグ用ツールからみると、ある程度デバッグ情報が異なるものに対しても、同じインタフェースでデバッグ情報にアクセスが可能となり、プロセッサごとに異なるデバッグ情報に対応して、個別にデバッグ用ツールを開発する必要がない。

（３）さらに、各種デバッグフォーマットに対応する部分（システム対応サーバ側の個別部）をモジュール化することで、対応するデバッグフォーマットの種類を随時増やすことができる。

【００７２】４．変形例

本発明は、上記実施形態に限定されるものではなく、例えば以下のように種々の変形が可能である。

（１）本実施形態においては、デバッグ情報を生成するコンパイラとして、ＣコンパイラまたはＣ＋＋コンパイラを念頭において説明したが、個別のアクセスルーチンを追加することで、その他のプログラム言語（例えばＪａｖａ等）のコンパイラにも対応可能である。

（２）本実施形態においては、仮想サーバが１台の場合を例に挙げたが、多数のクライアントからの処理が集中するような場合には、仮想サーバとしての機能を有するミラーサーバを設けて、このミラーサーバに適宜処理を分散させることも可能である。

【００７３】

【発明の効果】以上詳細に説明したように、この発明によれば、プログラムのデバッグ作業に際し、リモートログインせずにローカル端末から共通のインタフェースでシンボル情報を取得することができる。

【図面の簡単な説明】

【図１】本実施形態にかかるデバッグシステムの概要を示した構成図である。

【図２】同上デバッグシステムの具体的なシステム構成について示したものである。

【図３】クライアント、仮想サーバ、各システム対応サーバ間の動作シーケンスについて示したものである。

【図４】同上デバッグシステムを用いてデバッグ作業を行う際のユーザ側を中心とした処理の流れについて示したものである。

【図５】システム対応シンボリックサーバの機能構成について示したものである。

【図６】ＥＬＦ／ＤＷＡＲＦを用いて行番号アドレス変換が行われる際のシーケンスを示したものである。

【図７】ファイルフォーマットやデバッグフォーマットに定義されていない情報要素にアクセスする際の機構について示したものである。

【図８】ファイルフォーマットとデバッグフォーマットとの関係について示したもので、（ａ）はデバッグ作業面からみた双方の関係について示し、（ｂ）は具体的な各フォーマットの相互間の関係を示したものである。

【図９】ＥＬＦファイルフォーマットの構成を示したものである。

【図１０】ＥＬＦファイルフォーマットにおける各情報

要素間の依存関係について示したものである。

【図１１】ＤＷＡＲＦデバッグフォーマットにおける各情報要素間の依存関係について示したものである。

【図１２】ソースプログラムの一例を挙げ、特にそのソースプログラムに記述されたローカル変数について示したものである。

【図１３】クライアントからシステム対応サーバにデバッグ情報取得要求が発行されてから、実際にデバッグ情報が返送されるまでのシーケンスの一例を示したものである。

【図１４】ＤＷＡＲＦにおけるシンボリック情報取得の際の状況を示したものである。

【図１５】ＤＷＡＲＦにおけるシンボルテーブルの構成を示したもので、（ａ）はユニット情報を、（ｂ）はグローバル関数情報を、（ｃ）はローカル変数情報を示している。

【図１６】従来の遠隔デバッグシステムの構成を示したものである。

【符号の説明】

１　　ソースファイル

２　　コンパイラ

３　　実行形式ファイル

４　　デバッグ情報

ＶＳ　　仮想サーバ

ＷＳ　　監視用サーバ

ＳＭ　　サーバマシン

ＡＬ　　アクセス用ライブラリ

ＥＮＥＴ　イーサネット（登録商標）

ＦＡ　　ファイルフォーマットアクセス部

ＤＡ　　デバッグフォーマットアクセス部

ＳＳ（Ｓα，Ｓβ，Ｓγ，・・・）　システム対応（シンボリック）サーバ

Ｃ（Ｃ１，Ｃ２，Ｃ３，・・・）　クライアント

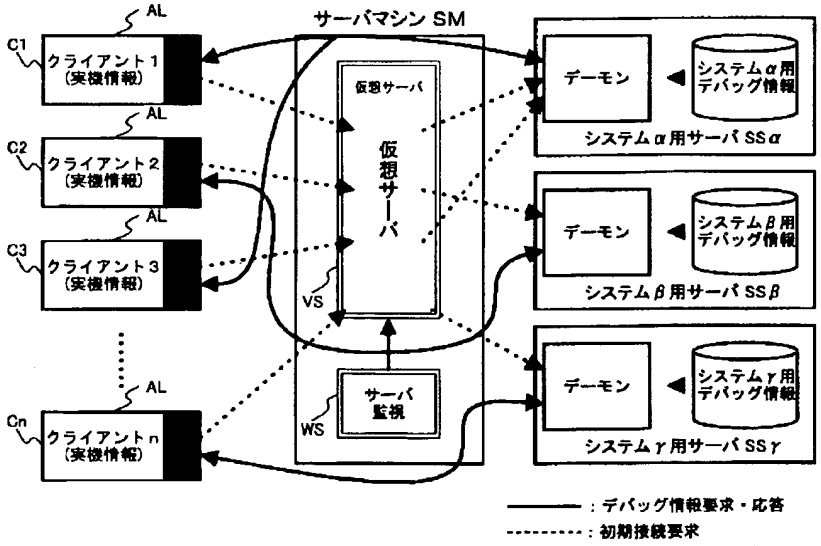
Ｅ（Ｅα，Ｅβ，Ｅγ，・・・）　プロセッサごとの個別部

ＬＭ（ＬＭα，ＬＭβ，ＬＭγ，・・・）　ロードモジュール

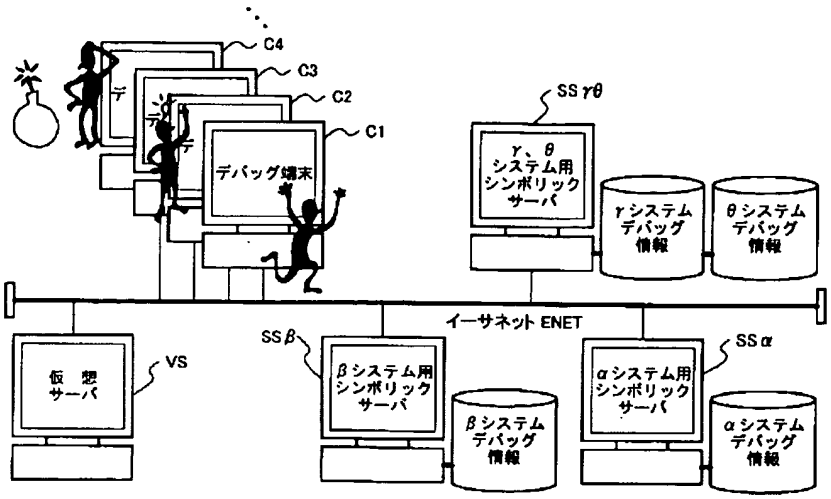
ＥＬ１～ＥＬ３１　ＥＬＦファイルフォーマット構成要素

ＤＷ１～ＤＷ３５　ＤＷＡＲＦファイルフォーマット構成要素

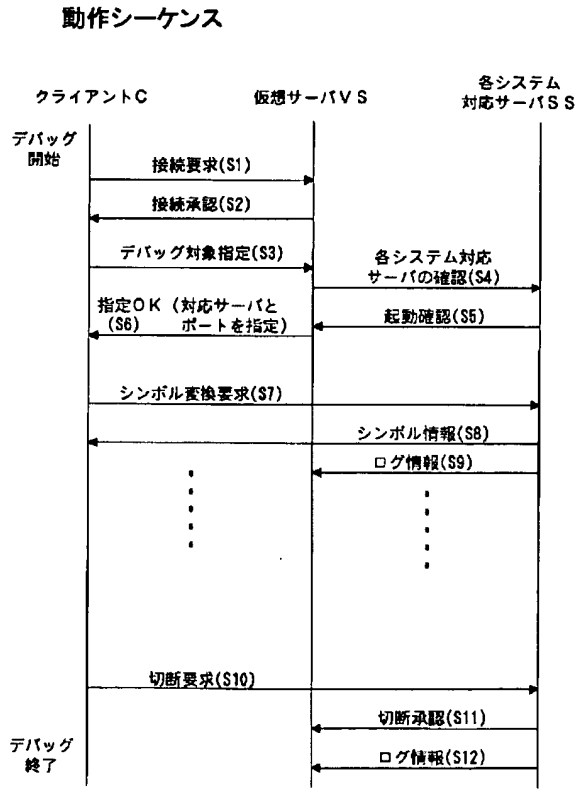
【図１】



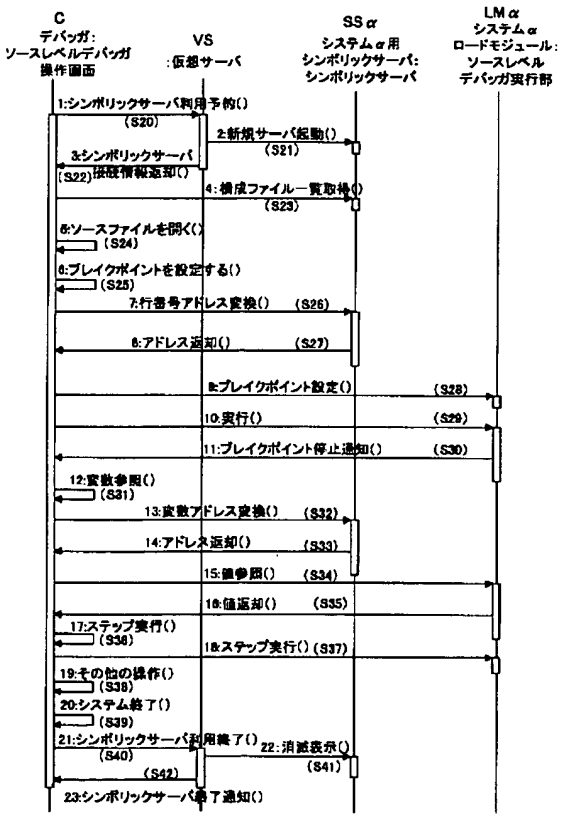
【図２】


C4
C3
C2
C1
デバッグ端末
SS γθ
γ、θ
システム用
シンボリック
サーバ
γシステム
デバッグ
情報
θシステム
デバッグ
情報
イーサネット ENET
仮想
サーバ
VS
SS β
βシステム用
シンボリック
サーバ
βシステム
デバッグ
情報
SS α
αシステム用
シンボリック
サーバ
αシステム
デバッグ
情報

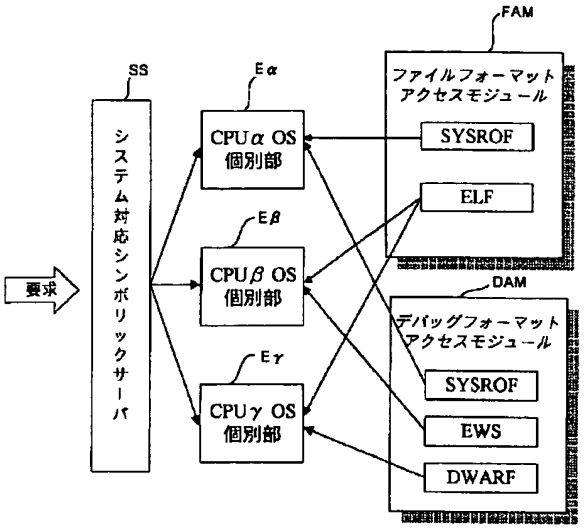
【図３】


動作シーケンス
クライアントC
仮想サーバVS
各システム
対応サーバSS
デバッグ
開始
接続要求(S1)
接続承認(S2)
デバッグ対象指定(S3)
各システム対応
サーバの確認(S4)
指定OK（対応サーバと
(S6)　ポートを指定)
起動確認(S5)
シンボル変換要求(S7)
シンボル情報(S8)
ログ情報(S9)
切断要求(S10)
切断承認(S11)
デバッグ
終了
ログ情報(S12)

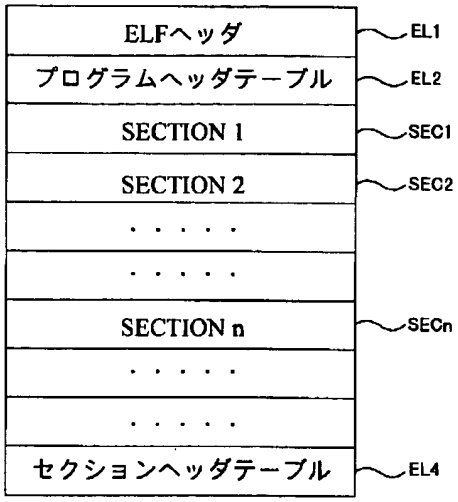
【図４】


C
デバッガ:
ソースレベルデバッガ
操作画面
VS
:仮想サーバ
SS α
システムα用
シンボリックサーバ:
シンボリックサーバ
LM α
システムα
ロードモジュール:
ソースレベル
デバッガ実行部
1:シンボリックサーバ利用予約()
(S20)
2:新規サーバ起動()
(S21)
3:シンボリックサーバ
(S22)接続情報返却()
4:構成ファイル一覧取得()
(S23)
5:ソースファイルを開く()
(S24)
6:ブレイクポイントを設定する()
(S25)
7:行番号アドレス変換()　(S26)
8:アドレス返却()　(S27)
9:ブレイクポイント設定()　(S28)
10:実行()　(S29)
11:ブレイクポイント停止通知()　(S30)
12:変数参照()
(S31)
13:変数アドレス変換()　(S32)
14:アドレス返却()　(S33)
15:値参照()　(S34)
16:値返却()　(S35)
17:ステップ実行()
(S36)
18:ステップ実行()(S37)
19:その他の操作()
(S38)
20:システム終了()
(S39)
21:シンボリックサーバ利用終了()
(S40)
22:消滅表示()
(S41)
(S42)
23:シンボリックサーバ終了通知()

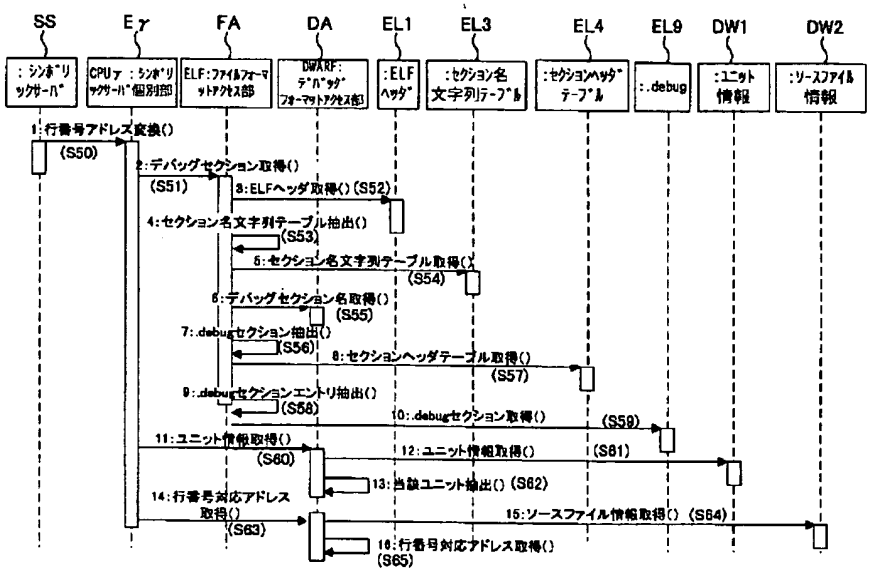
【図５】


FAM
SS
Eα
ファイルフォーマット
アクセスモジュール
CPUα OS
個別部
SYSROF
ELF
Eβ
CPUβ OS
個別部
DAM
デバッグフォーマット
アクセスモジュール
SYSROF
EWS
DWARF
Eγ
CPUγ OS
個別部
要求
システム対応シンボリックサーバ

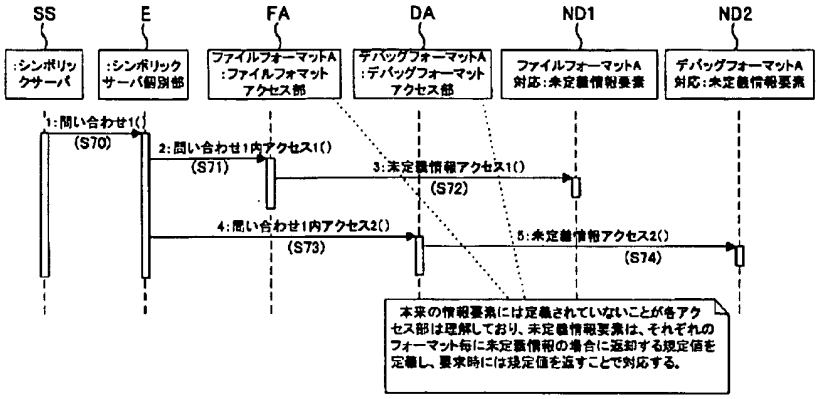
【図９】


ELFヘッダ
EL1
プログラムヘッダテーブル
EL2
SECTION 1
SEC1
SECTION 2
SEC2
SECTION n
SECn
セクションヘッダテーブル
EL4

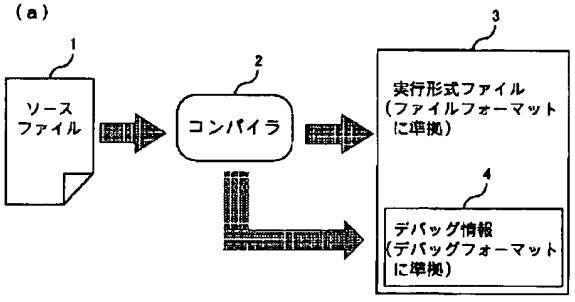
【図６】

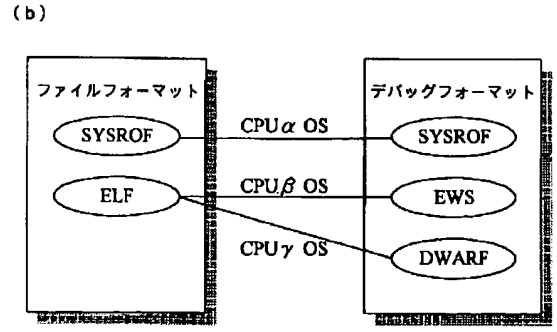

SS
Eγ
FA
DA
EL1
EL3
EL4
EL9
DW1
DW2
1:行番号アドレス変換()
(S50)
2:デバッグセクション取得()
(S51)
3:ELFヘッダ取得() (S52)
4:セクション名文字列テーブル抽出()
(S53)
5:セクション名文字列テーブル取得()
(S54)
6:デバッグセクション名取得()
(S55)
7:.debugセクション抽出()
(S56)
8:セクションヘッダテーブル取得()
(S57)
9:.debugセクションエントリ抽出()
(S58)
10:.debugセクション取得()
(S59)
11:ユニット情報取得()
(S60)
12:ユニット情報取得()
(S61)
13:当該ユニット抽出() (S62)
14:行番号対応アドレス
取得()
(S63)
15:ソースファイル情報取得() (S64)
16:行番号対応アドレス取得()
(S65)

【図７】

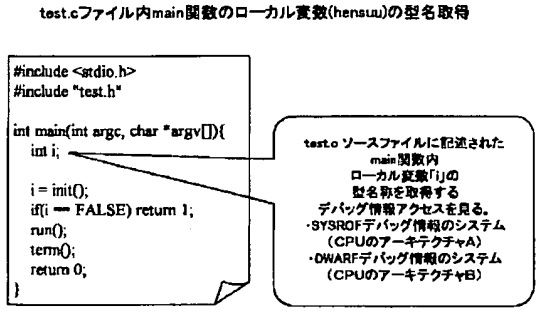

SS
E
FA
DA
ND1
ND2
:シンボリックサーバ
:シンボリックサーバ個別部
ファイルフォーマットA :ファイルフォーマットアクセス部
デバッグフォーマットA :デバッグフォーマットアクセス部
ファイルフォーマットA 対応:未定義情報要素
デバッグフォーマットA 対応:未定義情報要素
1:問い合わせ1()
(S70)
2:問い合わせ1内アクセス1()
(S71)
3:未定義情報アクセス1()
(S72)
4:問い合わせ1内アクセス2()
(S73)
5:未定義情報アクセス2()
(S74)
本来の情報要素には定義されていないことが各アクセス部は理解しており、未定義情報要素は、それぞれのフォーマット毎に未定義情報の場合に返却する規定値を定義し、要求時には規定値を返すことで対応する。

【図８】

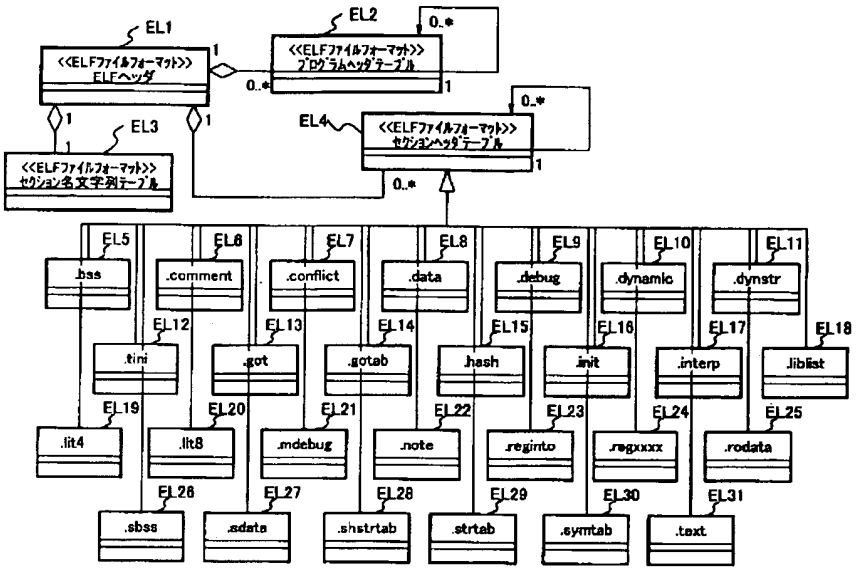

(a)
1
ソースファイル
2
コンパイラ
3
実行形式ファイル
(ファイルフォーマットに準拠)
4
デバッグ情報
(デバッグフォーマットに準拠)


(b)
ファイルフォーマット
SYSROF
ELF
CPUα OS
CPUβ OS
CPUγ OS
デバッグフォーマット
SYSROF
EWS
DWARF

【図１２】

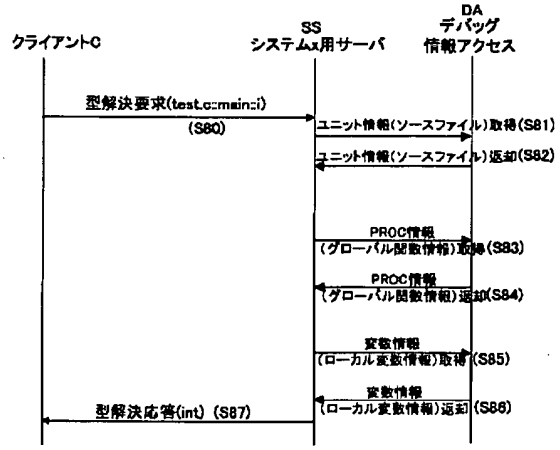

test.cファイル内main関数のローカル変数(hensuu)の型名取得
#include <stdio.h>
#include "test.h"
int main(int argc, char *argv[]){
int i;
i = init();
if(i == FALSE) return 1;
run();
term();
return 0;
}
test.o ソースファイルに記述されたmain関数内ローカル変数「i」の型名称を取得するデバッグ情報アクセスを見る。
・SYSROFデバッグ情報のシステム(CPUのアーキテクチャA)
・DWARFデバッグ情報のシステム(CPUのアーキテクチャB)

【図１０】



【図１３】

システム対応サーバの受け付けイメージ



【図１５】

(a)

DWARF
ユニット情報(ソースファイル)　DW1

DWARF:ユニット情報(ソースファイル)			
	属性	サイズ(byte)	型
1	ユニット名	n+1	string
2	使用言語	4	word
3	ユニット開始アドレス	4	address
4	ユニット終了アドレス	4	address
5	行番号情報オフセット	4	word
6	コンパイルディレクトリ	n+1	string
7	作成者	n+1	string

(b)

DWARF
PROC情報(グローバル関数情報)　DW9

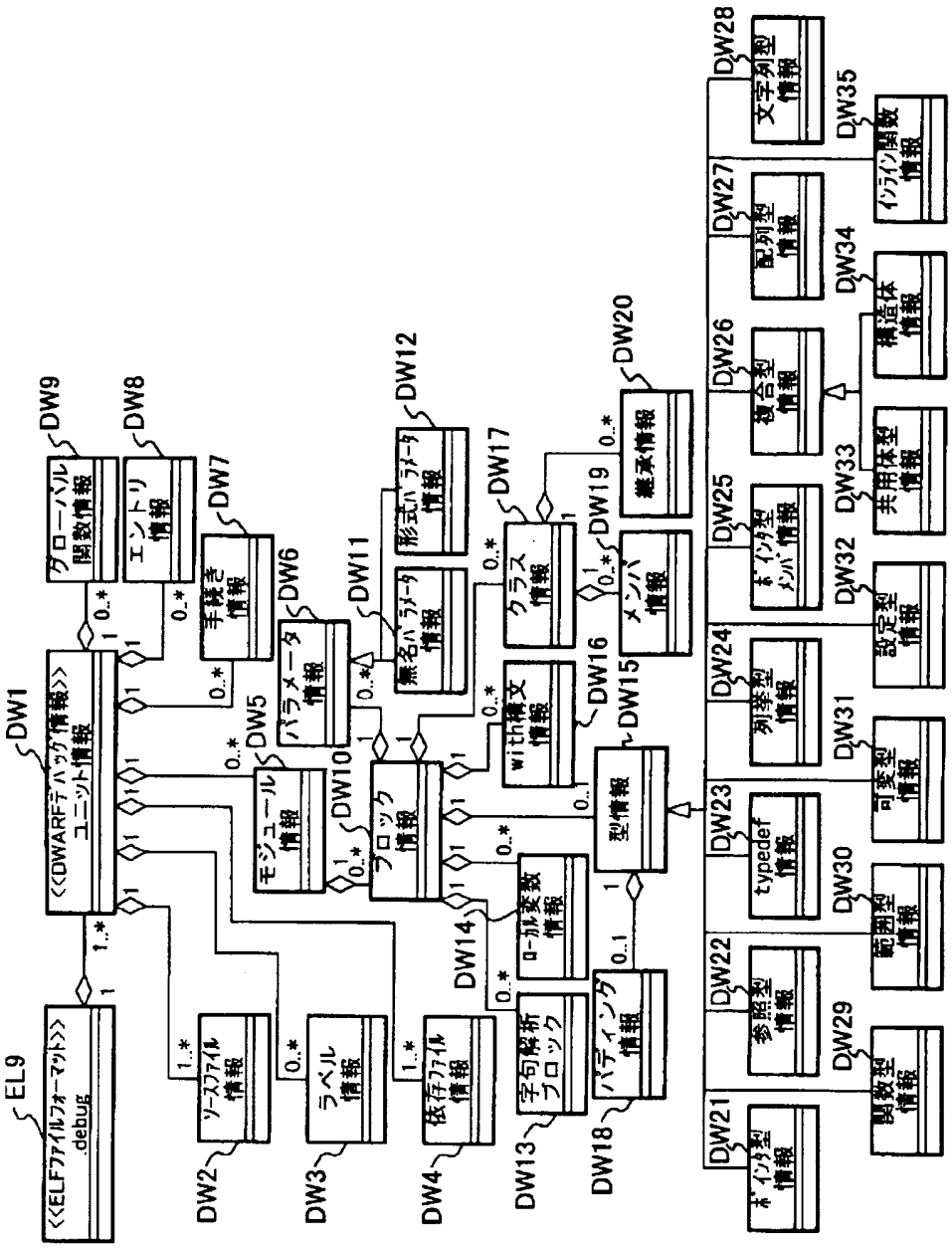
DWARF:PROC情報(グローバル関数情報)			
	属性	サイズ(byte)	型
1	関数名	n+1	string
2	基本型情報	2	word
3	修飾基本型情報		block2
4	ユーザ型情報	4	reference
5	修飾ユーザ型情報		block2
6	関数開始アドレス	4	address
7	関数終了アドレス	4	address

(c)

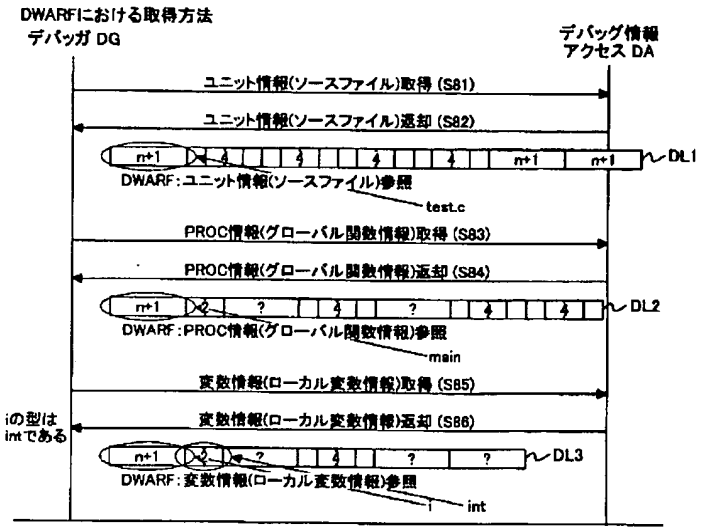
DWARF
変数情報(ローカル変数情報)　DW14

DWARF:変数情報(ローカル変数情報)			
	属性	サイズ(byte)	型
1	変数名	n+1	string
2	基本型情報	2	word
3	修飾基本型情報		block2
4	ユーザ型情報	4	reference
5	修飾ユーザ型情報		block2
6	ロケーション情報		block2

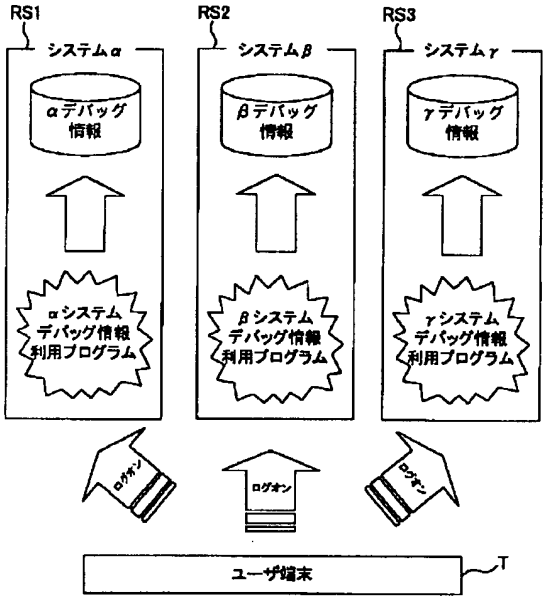
【図１１】


EL9
<<ELFファイルフォーマット>>
.debug
DW1
<<DWARFデバッグ情報>>
ユニット情報
DW2
ソースファイル情報
DW3
ラベル情報
DW4
依存ファイル情報
DW5
モジュール情報
DW6
パラメータ情報
DW7
手続き情報
DW8
エントリ情報
DW9
グローバル関数情報
DW10
ブロック情報
DW11
無名パラメータ情報
DW12
形式パラメータ情報
DW13
字句解析ブロック
DW14
ローカル変数情報
DW15
型情報
DW16
with構文情報
DW17
クラス情報
DW18
バディング情報
DW19
メンバ情報
DW20
継承情報
DW21
ポインタ型情報
DW22
参照型情報
DW23
typedef情報
DW24
列挙型情報
DW25
ポインタ型メンバ情報
DW26
複合型情報
DW27
配列型情報
DW28
文字列型情報
DW29
関数型情報
DW30
範囲型情報
DW31
可変型情報
DW32
設定型情報
DW33
共用体型情報
DW34
構造体情報
DW35
インライン関数情報

【図１４】



【図１６】



フロントページの続き

(72)発明者　山田　洋一
東京都港区港南一丁目９番１号　エヌ・ティ・ティ・コミュニケーションウェア株式会社内

(72)発明者　神人　將彰
東京都港区港南一丁目９番１号　エヌ・ティ・ティ・コミュニケーションウェア株式会社内

(72)発明者　楯　武士
東京都港区港南一丁目９番１号　エヌ・ティ・ティ・コミュニケーションウェア株式会社内

Ｆターム(参考)　5B042 GA02 GA12 GC08 HH04 HH41
5B045 GG01 JJ49
5B085 AC09 BG07